What is claimed is:

1. A method for synchronizing redundant processing units which are clocked synchronously or asynchronously, comprising:

providing an identical instruction sequence to each of the redundant processing units;

assigning a module to each of the processing units;

monitoring transactions that are external to the processing units via the modules; and

achieving synchronization by placing the processing units in a wait state via the modules until the processing units have reached a current transaction.

2. The method according to Claim 1, further comprising transferring parameters by the modules via connections for synchronization of the processing units which are characteristic of the transactions.

3. The method according to Claim 2, wherein executing a read transaction comprises:

leaving a processing unit in the wait state until arrival of data to be read via the module associated with the respective processing unit;

sending the parameters of the read transaction to the module connected most directly with a transaction destination;

at the module connected most directly to the transaction destination, receiving and comparing the parameters from other modules and locally created parameters;

executing the read transaction and distributing the read data to the modules upon determining that the parameters match; and

at each module, forwarding the read data to the assigned processing unit and enabling continuation of instruction processing.

4. The method according to Claim 3, further comprising executing a data comparison to check the data integrity by reading data areas from main memories at regular intervals or on request and comparing the parameters of the read transactions, the comparison being made by at least one of the modules.

5. The method according to Claim 2, wherein the executing a write transaction comprises:

leaving a processing unit in the wait state until a write process is completed via the module associated with that processing unit;

sending the parameters of the write transaction to the module connected most directly with a transaction destination;

at the module connected most directly to the transaction destination, receiving and comparing the parameters from other modules and locally created parameters;

executing the write transaction and acknowledging the write process to the modules upon determining that the parameters match; and

at each module, enabling continuation of instruction processing for the assigned processing unit.

6. The method according to Claim 2, wherein external events are buffered, whereby stored external events are called in a

special operating mode of the processing units for processing by at least one execution unit of the processing units and the processing unit enters the operating mode in response to fulfillment of a condition that is pre-specified by instructions or fixed in advance, and continuation of instruction execution is delayed by the modules until the processing units have ended the special operating mode.

7. The method according to Claim 6, wherein a change to the special operating mode is made if comparator elements of the processing unit find a match between a counter element and register elements, whereby content of the register elements are specified by commands and are identical for the processing units and the counter element includes a number of instructions completed by the execution unit since the last change into the special operating mode.

8. The method according to Claim 7, wherein the external events routed to the processing units initiate an event handling routine which begins with the read transaction of an event vector, whereby the read transaction is executed by the module assigned to the processing unit by leaving the processing unit in the wait state until arrival of the data to be read and sending the parameters of the read transaction to the destination linked most directly to the transaction, whereby the module linked most directly to the transaction destination receives and compares the parameters of the other modules and locally created parameters and, if they match, executes the read transaction and distributes the data read to the modules, where the modules forward the read data to the assigned processing units and initiate continuation of the instruction execution.

9. The method according to Claim 1, further comprising providing a direct memory access for transmission of data from the memory to an input/output module through initiation of direct memory access by jobs generated by a processing unit being transferred to the input/output module by entry into a register.

10. The method according to claim 1, further comprising providing a direct memory access for transmission of data from an input/output module into memory, such that a descriptor generated by an input/output module is stored in memory and is read out by the processing units with a polling procedure,

reading a register in one of the modules by the processing units causing no more write transactions in the memory by input/output modules,

writing a last of the write transactions sent by the input/output modules by the modules into the memory of the processing units,

reading a memory location in the memory of the processing units for which a value shows completion of a direct memory access, and

reading or writing to the register or another register to permit write access to the memory by the I/O units.

11. The method according to claim 1, further comprising providing a direct memory access for transmission of data between input/output module and a memory,

reading a register in one of the modules by the processing units causing no more read transactions by the input/output modules permitted in the memory,

storing a descriptor generated by the processing units in the memory which can be read out by one or more input/output modules with a polling procedure,

reading or writing the register or another register to permit read access to the memory by the I/O units, and

reading a memory location in the memory of one or more input/output modules, for which the value indicates the beginning of a direct memory access.

12. The method according to claim 2, wherein fault handling is initiated by a module linked most directly to a transaction destination if a deviation from the parameters of the other modules and locally generated parameters are established.

13. The method according to Claim 12, wherein the fault handling stops the transaction to be executed and starts a routine for detection of the faulty unit, the isolation and recovery of which to re-establish the synchronicity.

14. The method according to Claim 12, wherein with N available processing units the error handling makes an N-M (M < N) out of N majority decision and deactivates a divergent processing unit.

15. The method according to Claim 2, wherein failures of individual processing units are detected such that for a transaction beginning with an earliest availability of the parameters at the module of a processing unit, error processing is initiated for processing units with parameters that do not arrive or arrive after expiry of a pre-specified time.

16. The method according to Claim 1, wherein at least one of the following transactions are used by the modules for synchronization of the processing unit:

non-cacheable memory transactions relating to a local memory assigned to a relevant processing unit,

input/output transactions for input/output modules,

memory-mapped input/output transactions for external registers, and

non-cacheable memory transactions relating to a common memory of processing units.

17. The method according to Claim 2, wherein at least one of the following parameters of transactions are transferred by the modules via connections for synchronization of the processing units:

input/output addresses,

memory addresses,

data to be transferred,

type of transaction,

a signature formed from the input/output addresses,

the memory addresses,

the data to be transferred, and

the type of transaction.

18. An arrangement to synchronize synchronously or asynchronously clocked processing units of redundant data processing systems, comprising:

at least two processing units for processing identical instruction sequences;

peripherals assigned to each of the processing units for saving and/or exchanging data; and

peripherals jointly usable by the processing units for saving and/or exchanging data; and

modules assigned to each of the processing units, the modules including a first unit to monitor transaction, a second unit to stop the associated processing unit until a cur-

rent transaction has been reached by the processor units, and a third unit to transfer parameters of the transactions to other modules.

19. The arrangement in accordance with Claim 18, wherein the processing units include:

at least one execution unit,

at least one Completed Instruction Counter to count instructions executed by an execution unit since a last change into a special operating mode,

at least one register element, for which contents can be specified by instructions or is able to be fixed, and

at least one comparator element to switch over the execution unit into the special operating mode in response to the completed instruction counter matching register element, whereby in special operating mode buffered external events which influence the processor modules to be routed to the processor modules are called by the processor modules.

20. The arrangement in accordance with Claim 18, wherein the modules include a fourth unit to synchronize the processing units, based on the following transactions:

non-cacheable memory transactions relating to a local memory assigned to a relevant processing unit,

input/output transactions for input/output modules,

memory-mapped input/output transactions for external registers, and

non-cacheable memory transactions relating to a common memory of processing units.

21. The arrangement in accordance with Claim 18, wherein the modules include a fifth unit to form the following parameters representative for transactions:

input/output addresses,

memory addresses,

data to be transferred,

type of transaction, and

a signature formed from at least one of the input/output addresses, the memory addresses, the data to be transferred, and the type of transaction.